VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Meranie teploty pomocou Arduina a následný prenos dát do OS Linux

Temperature Measuring with Arduino and Data Transmission to OS Linux

2019 Lukáš Kučák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Lukáš Kučák**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Měření teploty pomocí Arduina a následný přenos dat do OS Linux
Temperature Measuring with Arduino and Data Transmission to OS Linux

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem bakalářské práce je navrnout systém pro automatizované měření teploty pomocí senzorů v sytému Arduino a následný přenos naměřených dat do operačního systému Linux.

1. Popište základní typy senzorů.
2. Popište komunikační protokoly pro přenos dat.
3. Navrhněte automatizovaný systém pro měření teploty.
4. Zajistěte průběžné vykreslování grafu průběhu teploty v OS Linux.

Seznam doporučené odborné literatury:

[1] Hughes J.M. *Arduino: A Technical Reference: A Handbook for Technicians, Engineers, and makers.* O'Reilly 2016
[2] Gertz E., Justo P.D. *Environmental Monitoring with Arduino.* O'Reilly 2012

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Nevlud**

Datum zadání: 01.09.2017
Datum odevzdání: 12.07.2019

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

prof. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostravě 11. července 2019 .....

Súhlasím so zverejnením bakalárskej práce podľa požiadavku čl. 26, odst. 9 Študijného a skúšobného rádu pre štúdium v bakalárskych programoch VŠB-TU Ostrava.

V Ostravě 11. července 2019 .....

Abstrakt

V tejto bakalárskej práci sa porovnávaju senzory na meranie teploty pomocou Arduina. Boli tu použité štyri senzory. Pri každom z nich sú opísané ich technícké parametre priamo od výrobcov. Sú tu popísane ich výhody aj nevýhody. Je tu spísaný kód potrebný ku spusteniu merania a skript v nodejs k vytvoreniu aplikácie na servery, ktorá vykresľuje graf z nameraných hodnôt. Nakoniec je tu zhodnotenie jednotlivých senzorov z teoretického aj z praktického hľadiska a vytvorenie automatizovaného systému.

Klíčová slova: Arduino, senzory, zbernice, kódy, grafy, meranie , teplota

Abstract

In this bechelor work there is comparism of senzors for measuring the temperature using Arduino. There were four senzors used. For each senzor there are described technical paramters from manufacturers.Then there are discribed their benefits and cons. There is writen code needed for starting the measuring and script in nodejs for creation an aplication on server which draws graf from measured data. At the end there is evaluation of each senzor from theoretical and pratical view and creation of automatized system.

Key Words: Arduino, senzors, buses, codes, graphs, measuring, temperature

Obsah

Seznam použitých zkratek a symbolů

A/D	–	Analóg/Digitál
PTC	–	Pozitívny Teplotný Koeficient
NTC	–	Negatívny Teplotný Koeficient
SDA	–	Serial Data
SCL	–	Serial Clock
GND	–	Ground
ROM	–	Read Only Memory
USB	–	Universal Serial Bus
NRZI	–	Non Return To Zero

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Meranie teploty je jedna z každodenných činností. Teplota sa meria všade a takmer na všetkom. Môže sa merať teplota vzduchu, vody či iných tekutín, ale taktiež pevných látok, čo je dôležité v mnohých ohľadoch. Najdôležitejšie prípady sú, aby sa predišlo prehrievaniu alebo podchladeniu či už zariadení alebo aj človeka. Preto je podstatné, aby bola teplota odmeraná čo najpresnejšie.

V kapitole jedna budú teoreticky popísané všetky senzory na meranie teploty. Ďalej tu budú opísané ich schémy zapojenia aj s konkrétnymi obrázkami.

V druhej kapitole budú teoreticky opísané všetky zbernice použité na prenos dát či už medzi Arduinom a senzormi na meranie teploty alebo medzi Arduinom a počítačom.

V tretej kapitole budú opísané všetky kódy pre jednotlivé senzory potrebné na spustenie samotného merania teploty. Ďalej tu bude popísaný nodejs skript, ktorý sa bude spúšťať v termináli v operačnom systéme Linux alebo aj Windows. Tento skript bude prijímať dáta zo sériovej linky od Arduina zo všetkých spustených senzorov a ukladať ich do sqlite databázy. Tieto dáta sa budú rozdeľovať podľa identifikátora, ktorý má každý senzor priradený. Dáta sa budú ukladať v reálnom čase ako desatinné číslo. Potom tento nodejs skript spustí aplikáciu na servery, ktorá bude obsahovať boxy na zadanie času od kedy do kedy sa vykreslí graf s uloženými hodnotami v tejto dobe v sqlite databáze. Tento graf bude možné priblížiť alebo oddialiť, taktiež sa pohybovať zo strany na stranu a ukázať teplotu v danom konkrétnom bode merania.

V štvrtej kapitole budú ukážky výslednej aplikácie s obrázkami konkrétnych grafov vykreslených na servery a ich popis. V závere bude zhrnutie celej práce so zistenými výsledkami a pozorovaniami.

2 Teoretický popis jednotlivých častí

2.1 Platforma Arduino

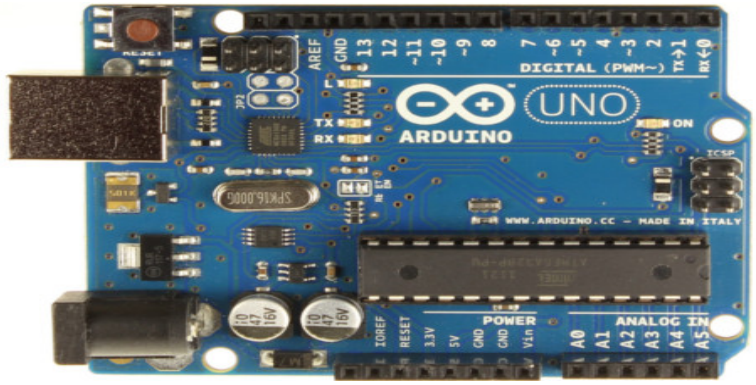
Najlepší popis platformy Arduino je, že je to mikrokontrolér a bol zámerne vyvinutý ľudmi, ktorí nie sú elektrikári, inžinieri ani programátori. Veľké výhody sú, že Arduino je pomerne lacné, multiplatformové a dosť jednoduché na programovanie. Jeho hardware aj software sú open-source. Ďalšie výhody sú, že Arduino je dosť malé a na svoju cenu a velkosť aj veľmi výkonné. Programátori, dizajnéri, samouci, ale aj výtvarníci využívajú silu a jednoduchosť tejto platformy na vytvorenie všetkých druhov inovatívnych zariadení, ako napríklad interaktívne senzory, umenie či hračky.[1]

2.1.1 Využitie

Arduino doska získaká údaje od rôznych snímačov a senzorov. Napríklad snímač osvetlenia,vzdialenosti, senzor teploty alebo len obyčajné tlačítko . Na základe týchto údajov ovláda nejaké výstupy. Ako príklad je možné si zobrať rozsvietenie LEDky, zapnutie svetla alebo motora, či iný fyzický výstup. Aby doska vykonávala to, čo potrebujeme, musí sa vytvoriť program pre Arduino mikrokontrolér. Na to sa využije programovací jazyk Arduino , ktorý je založený na jazyku Wiring a Arduino software alebo inak povedané IDE, založené na prostredí Processing.[2]

2.1.2 Zloženie

Základná Arduino doska je veľmi jednoduchá a dá sa vyrobiť aj svojpomocne alebo sa môžu súčiastky pospájať na kontaktnom poli. Arduino je prakticky tvorené z mikrokontroléra, kryštála, napájacieho zdroja, ktorý má obvykle 5 Voltov a prevodníka pre komunikáciu s počítačom. Podľa požadovanej funkcie, napríklad na ovládanie motora,displeja alebo bezdrátového modulu vašeho Arduino projektu je možné využiť širokú škálu rozširujúcich dosiek pre Arduino. Tieto dosky označujeme ako Arduino Shieldy. Dnes je ich veľké množstvo druhov.



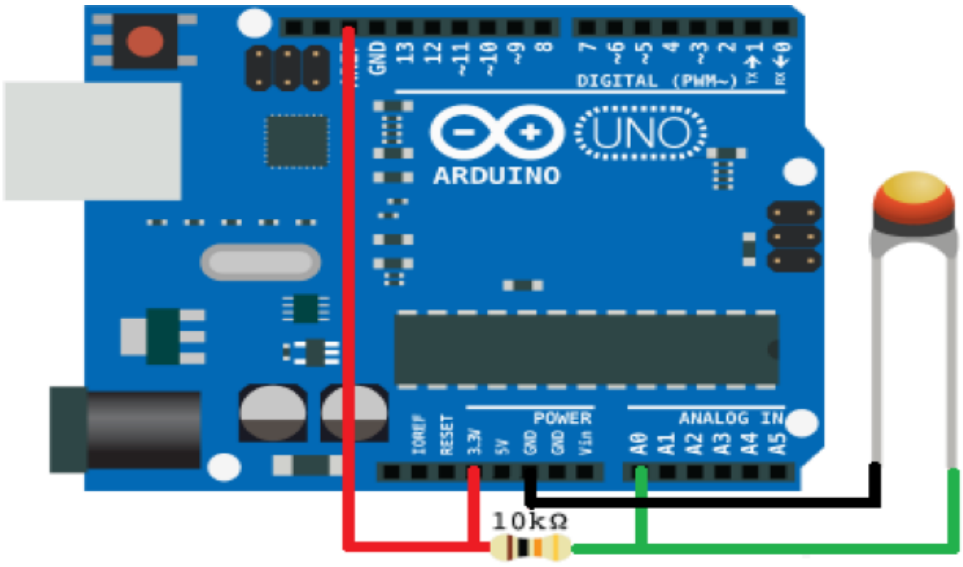
Obrázek 1: Arduino UNO
Prebrané z:[3]

2.2 Termistor

Meranie teploty je jedna z najbežnejších úloh. Teplota sa meria pomocou takzvaných termistorov. Ide o rezistory, ktoré v závislosti od teploty menia svoj odpor podľa presne definovanej charakteristiky. Termistory môžu mať buď pozitívny alebo negatívny teplotný koeficient. Pozitívny teplotný koeficient alebo skrátene PTC znamená, že so zvýšujúcou sa teplotou odpor termistoru rastie. U negatívneho teplotného koeficientu alebo skrátene NTC naopak odpor so zvýšujúcou teplotou klesá. Závislosť odporu na teplote není ani u jedného typu lineárna, ale v prípade pozitívneho teplotného koeficientu je výrazne priamejšia ako u negatívneho teplotného koeficientu. A taktiež termistor s negatívnym teplotným koeficientom má omnoho väčšiu citlivosť. Jednotlivé varianty ako aj u PTC tak aj u NTC termistorov sa rozlišujú podľa odporu pri teplote 25 stupňov Celzia. Veľmi obľúbeným PTC termistorom je PT100 s odporom 100 Ohm pri teplote 25 stupňov Celzia a u NTC sú najčastejšie hodnoty odporu 10 kilo Ohm a 100 kilo Ohm.[4]

2.2.1 Zapojenie

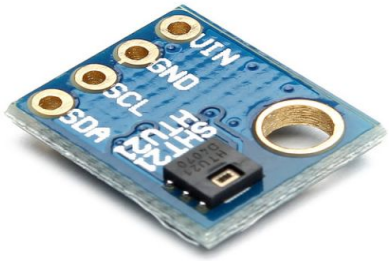
Termistor sa zapojí ako spodná časť napäťového deliča zatiaľ čo do hornej časti vložíme odpor o hodnote ako referenčný odpor termistora. Rezistor môžeme zapojiť ako na 5 tak aj na 3,3 Voltov. Konkrétna hodnota napätia nemá na meranie podstatný vpliv, avšek iba v prípade, že je táto hodnota súčasne referenčným napätím A/D prevodníka. Pri volbe 3,3 Voltov bude cez NTC pretekať menší prúd a samovolné ohrievanie termistoru meracím prúdom bude tak nižšie.



Obrázek 2: Schéma Termistoru
Prebrané z:[5]

2.3 HTU21D

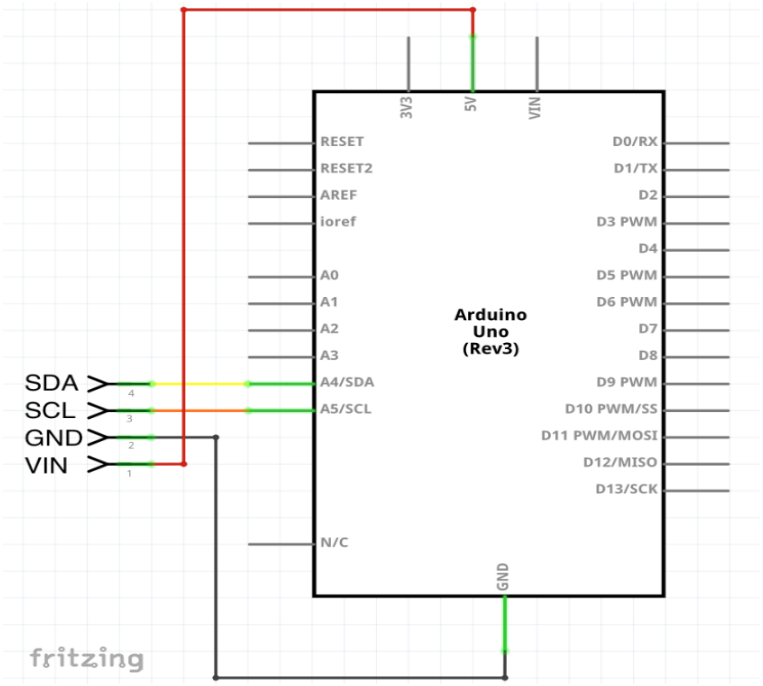
Senzor HTU21D je vstupný modul, ktorý umožňuje s pomocou Arduina merať teplotu a vlhkosť v okolí. Tento senzor dokáže merať teplotu v rozsahu od -40 až po +125 stupňov Celzia. Čo sa týka presnosti merania, tak je to presnosť typicky +-0,3 stupňov Celzia a najviac presné je meranie v rozsahu 5 až 60 stupňov Celzia. Napájacie napätie pre tento modul je výrobcom doporučené v rozsahu 3,3 až 5 Voltov. Odoberaný elektrický prúd je veľmi nízky. Ak je tento senzor zapojený, ale nemeria teplotu tak dosahuje maximálnej hodnoty 140 nanoAmpér a pri meraní dosahuje hodnotu maximálne 0,5 miliAmpér. Ku všetkým spomínaným výhodám treba taktiež dodať veľkosť celého modulu, ktorá je 12 x 10 mm. Senzor má tiež možnosť montážného otvoru.[6]



Obrázek 3: Senzor HTU21D
Prebrané z:[7]

2.3.1 Zapojenie

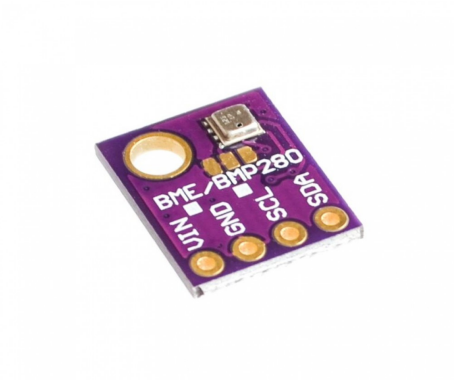
Pre úspešné prepojenie senzoru HTU21D s Arduinom stačí prepojiť 4 vodiče. Spojí sa SDA na module s pinom A4 na Arduino doske, ďalej je nutné prepojiť SCL na senzore s pinom A5 na Arduine. Potom už len stačí prepojiť GND na oboch pre uzemnenie a VIN na senzore s 3,3 alebo 5 Voltovým vstupom na Arduine. Komunikácia medzi Arduinom a senzorom prebieha po zbernici I2C, takže je nutné využiť pre dátové piny SCL a SDA odpovedajúce I2C piny na Arduino doske.



Obrázek 4: Schéma zapojenia na I2C
Prebrané z:[6]

2.4 BMP280

Modul pre meranie teploty, vlhkosti a biometrického tlaku obsahuje precízny senzor BMP280. Tento meriaci senzor od firmy Bosch komunikuje cez rozhranie I2C a zvláda komunikovať rýchlosťou až 3,4 MHz. Čo sa týka rozsahu na používanie a meranie senzoru BMP280, tak je to až od -40 po +85 stupňov Celzia. Presnosť merania je potom už len +-1 stupeň Celzia. Pre napájanie modulu so senzorom BMP280 môžeme použiť rozsah napájacieho napätia 1,8 až 5 Voltov. Schéma aj popis zapojenia je rovnaký ako u senzoru HTU21D.[9]



Obrázek 5: Senzor BMP280
Prebrané z:[9]

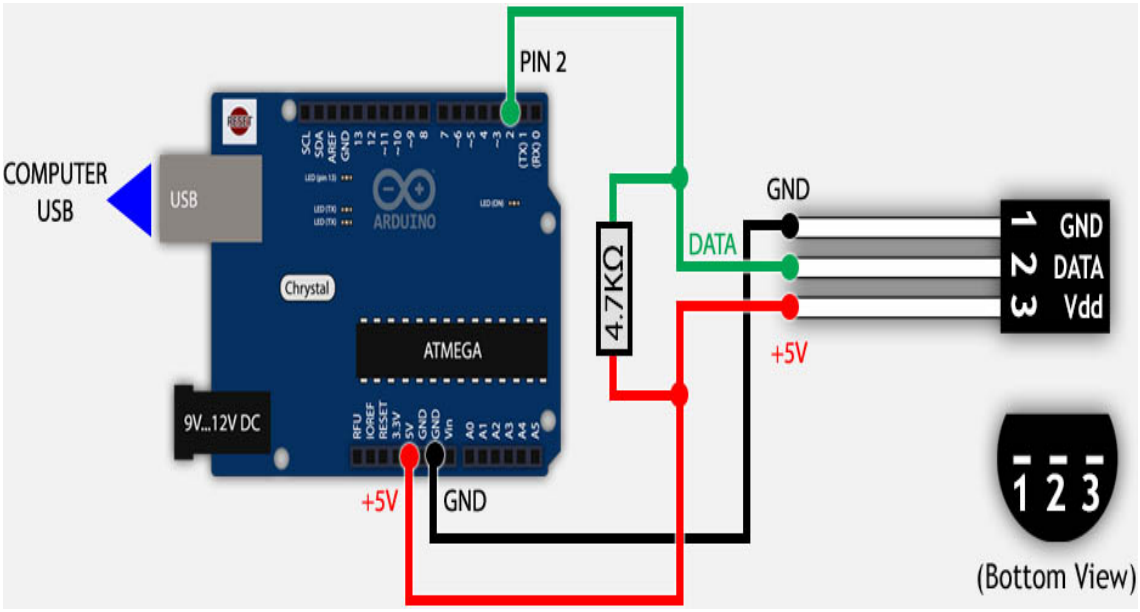
2.5 DS18B20

Teplotný senzor alebo taktiež čidlo DS18B20 od firmy Maxim, ktorá pred tým bývala Dallas je v Arduino komunite veľmi obľúbené. Za veľmi dobrú cenu umožňuje merať teplotu v rozsahu od -55 až po +125 stupňov Celzia, pričom v rozsahu -10 až +85 stupňov Celzia má garantovanú presnosť +-0,5 stupňa Celzia. Ďalšou výhodou je taktiež možnosť zakúpiť tento teplotný senzor buď v puzdre TO-92, ktoré sa veľkosťou podobá obyčajným tranzistorom, alebo aj vodotesnú variantu. Pri tejto vodotesnej variante je senzor v nerezovej tyčinke. Pre komunikáciu s Arduino doskou je u čidla DS18B20 využívaná zbernica OneWire, ktorá využíva iba jeden komunikačný pin a toto čidlo taktiež podporuje parazitný režim, kde pre spojenia senzoru s Arduino doskou stačí využiť len dva drôty. Teplotné čidlo DS18B20 je veľmi obľúbené hlavne pre svoju nízku cenu a ľahkú použiteľnosť. Umožňuje merať teplotu s dostatočnou presnosťou pre domáce použitie a je možné použiť prepájacie vodiče medzi čidlom a Arduinom dlhé až niekoľko desiatok metrov.[10]

Obrázek 6: Senzor DS18B20
Prebrané z:[11]

2.5.1 Zapojenie

Pre úspešné spojenie čidla DS18B20 s Arduino doskou stačí spojiť iba dva vodiče a jeden odpor o veľkosti 4700 Ohmov. Spojí sa GND na čidle s GND na doske. Ďalej sa prepojí krajný vodič čidla Vdd s 5 Voltovým pinom na Arduine, stredný vodič DATA môžeme spojiť s akýmkoľvek voľným dátovým pinom na doske, ale je nutné uviesť aj správny pin v programe pre Arduino dosku. Ešte je potrebné prepojiť odpor na jednom konci s vodičom DQ a na druhom konci s 5 Voltovým napätím. Pri vodotesnej variante sú farby vodičov nasledovné: červená = napájanie, čiarna = zem, zelená alebo biela = dátový pin DQ.



Obrázek 7: Schéma DS18B20
Prebrané z:[12]

3 Použité protokoly na prenos dát

3.1 One-Wire

Zbernica One-Wire, navrhnutá firmou Dallas Semiconductor umožňuje pripojiť niekoľko zariadení k riadiacej jednotke prostredníctvom dvoch vodičov. Zbernica má jeden riadiaci obvod, ktorý nazývame "master"a jeden alebo viac ovládaných zariadení, ktoré voláme "slave". Všetky obvody sú zapojené jednak na spoločnú zem, a taktiež paralelne na spoločný dátový vodič. Tento dátový vodič je pripojený cez odpor cca 5 kilo Ohm na napájacie napätie a mení tak zbernicu do log. 1.

Komunikáciu zahajuje vždy master reset pulsom. Najprv nastaví dátový vodič do log. 0, čiže ho uzemní a drží ho na tejto úrovni minimálne 480 mikrosekúnd. Potom zbernicu uvolní a počúva. Odpor zatiaľ vráti zbernicu späť do log. 1. Ak je na zbernicu pripojené nejaké One-Wire zariadenie, tak detekuje túto vzostupovú hranu a po časovom úseku 15 až 60 nanosekúnd stiahne zbernicu na 60 až 240 nanosekúnd ku log. 0.

Ak sa zariadenie správne ohlási, môže master začať vysielať a prijímať dáta. Dáta sú vysielané v takzvaných "time slotoch". Slot je dlhý 60 až 120 nanosekúnd a behom jedného slotu je vysielaný alebo prijímaný jeden bit informácie. Medzi jednotlivými slotmi musí byť minimálne 1 nanosekunda medzera, kedy je zbernica v pokoji.

Existujú štyri druhy slotov: Zápis 1, Zápis 0, Čítanie 1 a Čítanie 0. Zapisové sloty slúžia ku tomu, aby master vyslal data do zariadení. Zápis 1 prebieha tak, že master stiahne zbernicu k nule, minimálne na jednu nanosekundu a najneskôr na 15 nanosekúnd od začiatku ju opäť uvolní a ponechá uvolnenú. Zdvíhací odpor ju vtedy vytiahne ku log. 1. Zápis 0 je o niečo jednoduchší. Master stiahne zbernicu k 0 a ponechá ju tak po celý slot, teda minimálne 60 nanosekúnd. Zariadenie vzorkuje stav na dátovom vodiči zhruba 30 nanosekúnd po začiatku time slotu.

Sloty na čítanie master opäť inicializuje tým, že stiahne zbernicu k nule na minimálne 1 nanosekundu a následne ju znova uvolní. Po tomto zahájení môže vysielať 1 bit tým, že ponechá zbernicu v kľude alebo ju stiahne. Komunikácia prebieha po bajtoch a vždy sa vysiela bit 0, čiže najmenší ako prvý a bit 7 ako posledný.[13]

3.1.1 Viacej zariadení na jednej zbernici

Každé One-Wire zariadenie, u ktorého to dáva zmysel, napríklad iButtom, teplomery, prevodníky a podobne má v sebe pamäť ROM. Táto pamäť obsahuje 60 bitové unikátne číslo, pomocou ktorého je možné jednotlivé zariadenia na zbernici od seba navzájom odlíšiť. Pomocou tohoto čísla je každé zariadenie jednoznačne identifikovateľné. Komunikácia je potom zložitejšia, pretože po "RESET"pulze je potrebné vyslať príkaz "Match ROM", potom 64 bitový kód zariadenia, s ktorým sa má pracovať a až nakoniec sa posiela príkaz.

3.2 I2C

I2C zbernica je zkratka, ktorá vznikla z IIC zbernice. Celým názvom je to teda Internal-Integrated-Circuit. Ako už názov napovedá, ide o internú dátovú zbernicu slúžiacu pre komunikáciu a prenos dát medzi jednotlivými integrovanými obvodnmi väčšinou v rámci jedného zariadenia. Vyvinula ju firma Philips približne pred dvaciatimi rokmi a od tej doby prešla niekoľkými vylepšeniami. V dnešnej dobe tento spôsob komunikácie podporuje rada integrovaných obvodov, nie len od firmy Philips. Ide predovšetkým o mikrokotroléry, sériové pamäti, inteligentné LCD, audio a video, analóg na digitál, digitál na analóg prevodníky a niektoré ďalšie digitálne riadené obvody. Hlavnou výhodou je, že obojsmerný prenos prebieha len po dvoch vodičoch. Ide o dáta "SDA", čiže "serial data"a hodiny "SCL", teda o "serial clock". To predovšetkým u mikrokotrolérov výrazne optimalizuje nároky na počet vstupne-výstupných pinov a celkovo zjednodušuje výsledné zapojenie.

Na jednu zbernicu môže byť pripojených viac integrovaných obvodov. V základnej verzii sú obvody adresované 7 botovo a v rozšírenej verzii 10 bitovo. To umožňuje pripojenie 128 respektíve 1024 čipov s rôznou adresou na jednu spoločnú zbernicu. V praxi sú tieto čísla však podstatne nižšie, pretože adresu čipu väčšinou nejde určiť plnými siedmimi, či desiatimi bitmy ale napríklad len troma. Niekedy sa nedá určiť vôbec, pretože je daná na pevno pre daný typ čipu. Takýchto čipov teda nemôže byť na jednej zbernici viac ako jeden.

Prenosová rýchlosť zbernice je pre väčšinu aplikácii dostatočná aj v základnej verzii, kde je frekvencia hodín 100 kHz. Vo vylepšených verziách to môže byť 400 kHz, alebo až 1 MHz, ale nie všetky integrované obvody túto verziu podporujú. Rýchlosť prenosu potom musí byť prispôsobená pochopiteľne najpomalšiemu čipu na zbernici. Oba vodiče musia byť implicitne v logickoej jednotke a to je zaistené pull-up rezistormi. Ich hodnoty majú hodnotu v radoch jednotiek kilo Ohmov. Čím vyššia komunikačná frekvencia, tým musia byť nižšie hodnoty týchto odporov.[14]

3.2.1 Princíp prenosu

Jeden z integrovaných obvodov, večšinou mikrokontrolér, je nastavený ako master a všetky ostatné obvody sú nastavené ako slave. Obvody sa dajú zapojiť aj ako takzvaný multi-master, kde je čip mastrov niekoľko. Master pri akomkoľvek prenosu generuje hodinový signál na vodič SCL. Keď jeden čip vysiela, prijímajú všetky ostatné a len podľa adresy určujú, či sú dáta určené im. Čip, ktorý chce vysielať alebo prijímať dáta musí najprv definovať adresu čipu, s ktorým chce komunikovať. Ďalej musí určit, či pôjde o príjem alebo o vysielanie. Teda musí určiť, či ide o čítanie alebo zápis. Toto určuje R/W, teda read/write bit, ktorý je súčasťou adresy.

3.2.2 Prenos prebieha kombinovaním nasledujúcich celkov

3.2.2.1 Stav pokoja Je zaistený logickými jednotkami na oboch vodičoch, master teda negeneruje hodinový signál a neprebieha žiadny prenos dát. Logické jednotky sú na oboch vodičoch

zaistené pull-up rezistormi. Pull-up rezistory sú rezistory medzi vodičom a napájacím napätím. Stav pokoja nastane aj pokiaľ sú vstupy obvodu master v stave vysokej impedancie, čiže odpojené.

3.2.2.2 Štart bit Zahajuje prenos alebo jeho ďalšiu časť. Je vygenerovaný tak, že sa zmení úroveň SDA z jedna na nula, zatiaľ čo je SCL v logickej jednotke.

3.2.2.3 Stop bit Ukončuje prenos. Je vygenerovaný podobne ako štart bit. Logická úroveň SDA sa zmení z nula na jedna, zatiaľ čo je SCL v logickej jednotke. Stop bit môže byť vygenerovaný len po nepotvrdení prenosu, teda ak sa prijme Ack v logickej jednotke.

3.2.2.4 Prenos dát Dáta sú prenášané po 1 Byte teda po 8 po sebe idúcich bitov od najvyššieho po najnižší. Pri prenose dát sa môže logická úroveň na SDA meniť len ak je SCL v logickej nule. Pri každom pulzu na SCL je prenesený jeden bit.

3.2.2.5 Potvrdzujúci bit Ack Tento bit slúži k potvrdeniu dát. Ack bit sa posiela rovnakým spôsobom ako by sa odoslal deviaty bit dát. Je tam ale rozdiel že ho generuje čip, ktorý prijímal, a nie ten ktorý dáta posielal. Ak teda prenos prebehol v poriadku tak odošle logickú nulu. Logická nula potvrdzujúceho bitu znamená taktiež to, že je prijímač pripravený na príjem ďalšieho bytu, ktorý nasleduje okamžite po ňom pri ďalšom pulze na SCL. Ak prenos dát zlyhal pošle logickú jednotku. Alebo ak má dôjsť ku ukončeniu prenosu, tak neodošle nič. Pull-up rezistor zaistí, že bude na SDA logická jednotka a Ack bit v logickej nule odošle vysielač.

3.2.3 Spoľahlivosť a konštrukčné riešenie

Predovšetkým pri vyšších prenosových rýchlostiach, respektíve hodinových frekvenciách,čiže pri 400 kHz a 1 Mhz alebo pri ďalších vodičoch SCL a SDA by mohlo pri nesprávnom návrhu plošného spoja alebo celej konštrukcie dochádzať ku rušeniu a chybám v prenose. Okrem potvrdzujúceho Act bitu není prenos nijako kontrolovaný. Ack bit potvrdzuje iba prenesenie každého Byte dát, ale nie to, či boli dáta prenesené správne. Preto je vhodné aby vodiče SDA a SCL boli čo najkratšie, a taktiež aby sa v ich blízkosti nevyskytovali výkonné alebo vysokofrekvenčné obvody. Taktiež je odporúčané vyvarovať sa vzniku zemných smyčiek.

3.3 USB

3.3.1 Komunikačný protokol USB

Na začiatok je potrebné uviesť pár zákadných informácii o univerzálne sériovej zbernici, alebo inak povedané USB. Ide o externú zbernicu, po ktorej sa dáta prenášajú sériovo, čiže poloduplexne. Univerzálna sériová zbernica používa dva signálové vodiče s využitím diferenčného kódovania. Vďaka tomuto spôsobu prenosu sa i za cenu zvýšenia počtu vodičov v prepájaných

kábloch dosiahla väčšia odolnosť voči rušivím okolním vplyvom, ako je napríklad elektromagnetický šum. Na fyzickej úrovni sú jednotlivé komunikujúce zariadenia prepojené systémom point to point. To znamená, že jeden kábel prepája vždy len dva susedné uzly. Okrem jedného riadiaceho uzlu, ktorý nazývame host controller a obecne väčšieho počtu koncových zariadení je možné na zbernicu zapojiť aj rozbočovače. Tieto rozbočovače, ktoré sa nazývajú aj "hubs"majú len jeden vstup a niekoľko výstupov. Na logickej úrovni sa však jedná o skutočnú zbernicu, pretože riadiaci uzol môže komunikovať s akýmkoľvek pripojeným zariadením a to bez ohľadu na to v akom mieste stromovej štruktúry sa toto zariadenie v skutočnosti nachádza.

Vzhľadom ku tomu, že existuje vždy jeden riadiaci uzol a rozbočovače majú len jeden vstup je zaistené, že sa v topologickej štruktúre vzájomne prepojených uzlov nevyskytujú smyčky. Vstupné a výstupné konektory na rozbočovačoch sú taktiež konštrukčne odlišené. Toto všetko vedie ku značnému zjednodušeniu komunikačného protokolu a k možnosti pripájať a odpájať periférne zariadenie bez ovplyvnenia činnosti ostatných uzlov. Celkový počet zariadení, ktoré sa môžu pripojiť ku koreňovému uzlu dosahuje hodnoty 127, pretože každému zariadeniu je priradená jedna sedembitová adresa. Jedna adresa z tohoto rozsahu je určená pre prenos špeciálného tokenu do uzlu, ktorému ešte nebola priradená žiadna adresa. Zariadenia, ktoré sú na túto zbernicu pripojené môžu prenášať dáta niekoľkými štandardizovanými rýchlosťami. Pôvodná norma označovaná ako USB 1.0 stanovovala dve rýchlosti. Základnú rýchlosť 1,5 MB za sekundu, ktorá bola označovaná ako Full Speed a nízku rýchlosť pre pomalé zariadenia pracujúce na rýchlosti 187,5 kB za sekundu, ktorá bola nazývaná Low Speed. V roku 2001 bola v norme USB 2.0 stanovená ešte vyššia rýchlosť,čiže High Speed, pri ktorej ide teoreticky dosiahnuť až hodnotu 60 MB za sekundu. Rýchlosť prenosu reálnych informácii je v skutočnosti niekedy aj podstatne nižšia, pretože sa okrem užitočných dát musia prenášať aj ďalšie informácie, ako napríklad adresy, riadiace tokeny a tak ďalej.[15]

3.3.2 Prenosové médium

Pre prepojenie dvoch komunikačných uzlov sa používa kábel so štyrmi vodičmi, ktorý má na prvý pohľad celkom neobvylkú štruktúru, predovšetkým v porovnaní s kábelmy používanými u iných typoch zberníc. Dva vodiče označené symbolmi Vbus a GND slúžia pre napájanie pripojených periferných zariadení, čo umožňuje zjednodušenú konštrukciu týchto zariadení. Môžu to byť napríklad myšky, klávesnice, čítačky pamäťových kariet, flash disky, mikrokontroléry a tak ďalej. Ďalšie dva vodiče, ktoré sa označujú symbolmi D+ a D- sú určené pre sériový poloduplexný prenos dát s využitím diferenčného kódovania. Vďaka tomuto sa dosiahlo vyššej necitlivosti na okolné elektrostatické pole aj na možnosť predĺženia prenosovej linky. Oba signálové vodiče sú tvorené krútenou dvojlinkou, ktorej impedancia by mala dosahovať hodnoty približne 90 Ohmov. Odchylka od tejto hodnoty môže byť maximálne 15 percent. U tejto zbernice nie je vyvedený samostatný hodinový signál. Není to nutné, pretože vďaka použitému spôsobu kódovania bitov metódou NRZI s bit shiffing sa prijímacie zariadenie môže s každým preneseným bajtom znovu synchronizovať. Je zaručené, že v jednom bajte vždy dôjde ku zmene dátového signálu.[16]

3.3.3 Detekcia rýchlostí zariadení a ukončenie prenosovej linky

Ako už bolo povedané na začiatku tejto kepitoly konektory umiestnené na oboch koncoch pripojovacích kabelov a samozrejme ich opaky zabudované do zariadení sú konštrukčne odlišné, a to aj preto aby bola zaručená stromová topológia vzájomne prepojených uzlov. Elektrické vlastnosti konektorov umiestnených v samotných uzloch sú odlišné. Na strane konektoru typu A, jedná sa buď o host controller, čo je samotný počítač alebo výstupný konektor rozbočovača je na každú dátovú linku, čiže aj na D+ a taktiež na D- pripojený rezistor o hodnote 15 kilo Ohmov, ktorého druhý koniec je uzemnený. To znamená, že pokiaľ nie je na tento konektor zapojené žiadne ďalšie zariadenie, tak sú na oboch signálových vodičoch nízké úrovne nula. Tento stav sa označuje ako SE0, alebo aj Single Ended Zero. Vzhľadom ku tomu, že je tento stav ľahko a za všetkých okolností detekovateľný používa sa pre mnoho účelov. Napríklad pre posielanie príkazu reset alebo ukončeniu packetu a tak ďalej. V prenášaných dátach sa nikdy nemôže objaviť.

Oproti tomu na strane USB zariadenia alebo na vstupu rozbočovača je na jednej dátovej linke umiestnený pull-up rezistor o odpore 1,5 kilo Ohmu pripojený na napájacie napätie 3,3 Volta. Pokiaľ sa teda zariadenie pripojí k host controlleru, vytvorí sa napäťový delič 1,5 kilo Ohmu až 15 kilo Ohmov a napätie na tomto signálovom vodiči sa zvýší na logickú hodnotu jedna, čiže cca 3 Volty. Podľa toho, na ktorý signálový vodič je tento pull-up rezistor umiestnený je určená prenosová rýchlosť zariadenia. Ak je logická úroveň vodiča D+ jedna ide o zariadenie zvládajúce normálnú rýchlosť, čiže Full speed. Avšak ak je naopak zdvyhnutá logická úroveň druhého vodiča D- na jedna ide o zariadenie ktoré pracuje len s nízkov rýchlosťou, čiže Low Speed. Väčšinou na nízkej rýchlosti fungujú zariadenia typu klávesnica, myš alebo joyistick.

Zariadenia pracujúce v režimu vysokej rýchlosti,inak nazývanej aj High Speed sú zapojené rovnako ako zariadenia pracujúce v režimu normálnej rýchlosti, čiže taktiež obsahujú pull-up rezistor o hodnote 1,5 kilo Ohm zapojený medzi napájacím napätím 3,3 Volta a linke D+. Behom resetu sa zariadenie identifikuje ako High Speed zariadenie a pokiaľ aj druhá strana komunikácie podporuje režim High Speed, tak je pull-up rezistor odpojený cez tranzistor, pretože obe linky musia byť kvôli diferenčnému kódovaniu vyvážené. Naopak v režimu vysokej rýchlosti sa aktivuje rezistor o hodnote 90 Ohmov zapojený medzi oba dátové vodiče, ktoré zaisťujú útlm odrazu. Rovnakú impedanciu má aj krútená dvojlinka. Jedna z príčin, prečo nejaké zariadenie nepracuje korektne napríklad pri jeho pripojení do lacnejšieho rozbočovača môže spočívať práve v tom, že není korektne odpojený pull-up rezistor po prechodu do režimu vysokej rýchlosti alebo naopak není pripojený terminátor, čiže rezistor zapojený medzi dátové vodiče.[16]

3.3.4 Logické úrovne a kódovanie prenášaných bitov

Pre zariadenia pracujúce v režime nízkej rýchlosti aj normálnej rýchlosti sa používajú zhodné napäťové úrovne na signálnych vodičoch. Pre nízku úroveň nula je stanovený rozsah 0 až 0,3 Volta. Pre vysokú logickú úroveň jedna je potom rozsah 2,8 až 3,6 Volta. Pre režim vysokej rýchlosti sú však napäťové úrovne celkom odlišné. Pri nízkej logickej úrovni nula je rozsah od

-10 až po 10 miliVoltov a pre logickú úroveň jedna je to 360 až 440 miliVoltov. Avšak tieto logické úrovne nula a jedna nereprezentujú priamo prenášané bity, pretože je použité trocha upravené kódovanie NRZI.

Dáta sú prenášané pomocou zmeny stavu dátových liniek. Rozoznávajú sa stavy K a J. Samotný prúd bitov je zakódovaný inverznou metódou NRZI, ktorej princíp je veľmi jednoduchý. Bit s hodnotou nula je reprezentovaný ako zmena stavu medzi J a K či naopak ako zmena stavu z K na J. Zatiaľ, čo bit s logickou hodnotou jedna znamená, že sa stav prenosovej linky nezmenil. To znamená, že linka ostane v pôvodnom stave. Toto však nie je všetko. Kvôli tomu, aby bola zaručená synchronizácia prenosových dát je v prípade, že sa v bitovom prúde vyskytne šesť za sebou idúcich jednotiek, tak sa naviac vloží bit s hodnotou nula, čo znamená že sa vynúti zmena stavu na linke. Túto zmenu môžu obidve komunikujúce strany využiť k vzájomnej synchronízacii hodin, pričom sa prijímač zosynchronizuje podľa vysielača. Ak by sa na prenosovej linke vyskytlo sedem za sebou idúcich jednotiek bitov s logickou hodnotou jedna tak je to považované za chybu.[16]

4 Automatizované meranie

V tejto časti je popísané ako spustiť automatizované meranie teploty. Na začiatku sa musí vybrať kód daný od výrobcu pre každý senzor s ktorým sa má teplota merať. Následne sa tento kód spustí pomocou skriptu v nodejs a po sériovej linke sa začnú posielať dáta. Všetky namerané údaje sa ukladajú do databázy. Táto databáza funguje ako server v počítači na ktorý je pripojené Arduino. Následne je napisaný javascript, ktorý z nameraných dát vykresľuje graf priamo na serveri a taktiež sa dá určiť dátum a čas od kedy do kedy chceme vidieť výsledné hodnoty v grafe.

4.1 Kódy pre jednotlivé senzory

Ak sa má teplota merať len pomocou jedného senzora je potrebné mať ku dánemu senzoru kód pre jeho spustenie. Podľa druhu senzora sa musia naimportovať všetky potrebné knižnice a ich inštancie, ktoré sú potrebné ku správnemu spusteniu merania. V tejto sekcii sú popísané kódy pre jednotlivé senzory, ktoré boli použité pri tvorbe tejto bakalárskej práce.

4.1.1 Program pre senzor HTU21D

Senzor HTU21D pracuje na zbernici I2C. Na začiatok sa musia v programe pripojiť konkrétne knižnice potrebné ku spusteniu merania teploty pomocou senzora HTU21D. Ďalej je potrebné inicializovať senzor z knižnice pod svojím názvom. Potom sa nastaví maximálna hodnota prenosu bitov za sekundu. Následne sa začne komunikácia so senzorom. V cykle sa potom načítavajú informácie do premennej. Ak všetko prebehlo v poriadku tak sa vypíšu namerané údaje a pošlú na sériovú linku aj s daným id senzora. Jednotka vo výpise označuje počet desatiných miest. Nakoniec sa už len určí dĺžka prestávky medzi meraniami v milisekundách. Tento kód je prevzatý od [6].

```
#include <Wire.h>
#include <SparkFunHTU21D.h>

HTU21D htu;

void setup() {
Serial.begin(9600);
htu.begin();
}
```

```
void loop() {
float teplota = htu.readTemperature();
Serial.print("1:");
Serial.print(teplota, 1);
Serial.println();
delay(60000);
}
```

Výpis 1: Program pre senzor HTU21D

4.1.2 Program pre senzor BMP280

Na začatie merania teploty pomocou senozra BM280 je potrebné pridať až tri knižnice. Ďalej je potrebné nastaviť adresu senzoru. Následne sa inicializuje senzor z knižnice. Potom sa nastaví maximálne množstvo bitov, ktoré je možné preniesť za sekundu. V tomto momente sa môže zahájiť komunikácia so senzorom BMP280. V prípade chyby je vypísaná hláška po sériovej linke a zastavený program. Potom sa načíta nameraná teplota zo senzoru do premennej a pošle sa na sériovú linku aj s id senzora. Nakoniec sa už len určí prestávka medzi meraniami v milisekundách. Tento kód je prevzatý od [9]

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#define BMP280_ADRESA (0x76)
Adafruit_BMP280 bmp;
void setup() {
Serial.begin(9600);
if (!bmp.begin(BMP280_ADRESA)) {
Serial.println("BMP280 senzor nenájdený, skontrolujte zapojenie!");
while (1); }
}
void loop() {
float teplota = bmp.readTemperature();
Serial.print("1:");
Serial.print(teplota);
Serial.println();
delay(10000);
}
```

Výpis 2: Program pre senzor BMP280

4.1.3 Program pre senzor DS18B20

Na začiatok je potrebné pripojiť dve knižnice. Ďalej je nutné nastaviť číslo vstupného pinu do ktorého je pripojený senzor. Následne je potrebné vytvoriť inštanciu oneWireDS z knižnice OneWire. Potom sa ešte vytvorí inštancia senzoryDS z knižnice DallasTemperature. Teraz sa ešte musí nastaviť maximálna prenosová rýchlosť bitov za minútu. Potom sa načítajú infromácie o teplote zo senzoru na danom pine. Následne sa už len pošlú namerané dáta po sériovej linke aj s id senzora. Na jednom pine môžeme postupne načítať všetky teploty pri zapojení viacerých čidiel pomocou zmeny čísla v zátvorke. Poradie ide podľa unikátnej adresy čidiel. Na koniec sa už len určí prestávka medzi meraniami v milisekundách. Tento kód je pravzatý od [10].

```
#include <OneWire.h>
#include <DallasTemperature.h>

const int pinCidlaDS = 4;

OneWire oneWireDS(pinCidlaDS);
DallasTemperature senzoryDS(&oneWireDS);

void setup(void) {
Serial.begin(9600);
senzoryDS.begin();
}

void loop(void) {
senzoryDS.requestTemperatures();
Serial.print("1:");
Serial.print(senzoryDS.getTempCByIndex(0));

delay(10000);
}
```

Výpis 3: Program pre senzor DS18B20

4.1.4 Program pre Termistor

Na začiatok je potrebné určiť analógový pin ku ktorému je termistor pripojený, referenčný odpor termistoru, teplotu pre referenčný odpor, beta faktor a hodnotu odporu rezistoru, ktorý je zapojený v sériovom obvode. Termistor nepotrebuje pripojiť žiadné knižnice. Taktiež je potrebné použiť externý pin AREP ako referenčné napätie pre A/D prevodník. Následne sa zmeria napätie na termistore a spraví sa konverzia zmeranej hodnoty na odpor termistoru. Potom sa vypočíta teplota podľa vzťahu pre beta faktor a výsledná vypočítana teplota sa pošle po sériovej linke aj s id senzora. Nakoniec sa už len vypíše výsledná teplota a určí sa prestávka v meraní v milisekundách. Tento kód je prevzatý od [4].

```
int termPin = 0;
int termNom = 10000;
int refTep = 25;
int beta = 3977;
int rezistor = 10000;

void setup(void) {
Serial.begin(9600);
analogReference(EXTERNAL);
}

void loop(void) {
float napeti;
napeti = analogRead(termPin);
napeti = 1023 / napeti - 1;
napeti = rezistor / napeti;
float teplota;
teplota = napeti / termNom;
teplota = log(teplota);
teplota /= beta;
teplota += 1.0 / (refTep + 273.15);
teplota = 1.0 / teplota;
teplota -= 273.15;
Serial.print("1:");
Serial.print(teplota);
delay(10000);
}
```

Výpis 4: Program pre Termistor

4.2 Kód pre viac senzorov naraz

Ak chceme pripojiť a merať teplotu s viacerími senzormi naraz je potrebné si na začiatku zistiť ich unikátnu adresu. Väčšina senzorov funguje na I2C zbernici. Každý senzor má svoju unikátnu adresu aby vedel rozlíšiť dáta, ktoré mu patria od tých čo patria ostatným senzorom. I2C zbernica totiž posiela dáta všetkým senzorom a len podľa adresy sa určije, ktorému senzoru tieto dáta patria. V ďalšej sekcii je popísaný jednoduchý Arduino kód pomocou ktorého sa dá určiť adresa každého senzora, ktorý je pripojený na I2C zbernicu. Následne pre spustenie merania už len stačí naimportovať všetky knižnice a vytvoriť ich inštancie podľa daného typu senzora. Nakoniec stačí spustiť meranie všetkých senzorov a poslať namerané hodnoty na sériovú linku kde sa uložia do databázy.

4.2.1 Zistenie I2C adresy

V prípade, že chceme zapojiť viacero senzorov na jednu I2C zbernicu potrebujeme vedieť ich unikátne adresy, podľa ktorých sa odlišujú. Tento jednoduchý Arduino sketch zistí adresy všetkých zariadení pripojených na I2C zbernicu Arduina. Tento I2C skener na adresy vracia hodnotu od Write.endTransmisstion, podľa ktorej zisťuje adresu konkrétného zariadenia pripojeného na I2C zbernicu. Ak nájde nejaké zariadenie vypíše jeho adresu v tvare 0x a jeho hexadecimálnu hodnotu v serial monitore. Nájdené adresy je potom možné skopírovať a použiť v iných kódoch. Tento kód sa dá samozrejme použiť na zistenie aj jednej adresy. Tento skener je prevzaný od [17].

```
#include <Wire.h>

void setup(void) {
  Wire.begin();
  Serial.begin(9600);
  while (!Serial);
}

void loop()
{
  byte error, address;
  int zariadenia;
  zariadenia = 0;

 for(address = 1; address < 127; address++ )
  {
    Wire.beginTransmission(address);
    error = Wire.endTransmission();
```

```
    if (error == 0)
    {
      Serial.print("I2C zariadenie nájdené na adrese 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      zariadenia++;
    }
    else if (error==4)
    {
      Serial.print("Neznáma chyba na adrese 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }

 if (zariadenia == 0)
    Serial.println("Žiadne zariadenie nebolo nájdené\n");

 delay(10000);
}
```

Výpis 5: Zistenie I2C adresy

4.3 Skript v nodejs pre automatizované meranie

Tento skript prijíma dáta z Arduina po sériovej linke a ukladá ich do sqlite databáze. Taktiež vypisuje prijaté dáta do konzole. Ukladá záznami od všetkým senzorov, z ktorých Arduino posiela po sériovej linke. Následne z týchto nameraných dát vytvorí graf na server porte 3000. Graf vytvára podľa zadaného dátumu a času merania. V aplikácii sa nachádzajú boxy na zadanie času od kedy do kedy chceme vidieť namerané hodnoty uložené v sqlite databáze. Ďalej je tam tlačítko reset pre vrátenie priblíženia do pôvodného stavu. Graf sa dá priblížiť aj oddialiť podľa potreby a po ukázani na jeho časť ukáže presnú nameranú teplotu v tej dobe. Tento skript je spustiteľný z terminálu v operačnom systéme Linux, ale aj Windows a je písaný v nodejs.

Na začiatku sa uložia požiadavky na serial port, sqlite databázu, http stránku v ktorej sa bude aplikácia zobrazovať a požiadavka na skupinkovanie dát z jednotlivých senzorov do polí podľa ich indexov. Ďalej sa zavolá aplikácia na internet a určí sa server port na ktorom bude táto aplikácia fungovať. Potom sa určí cesta, ktorá otvorí aplikáciu po zadaní príslušného portu na servery. Ďalej sa nastaví používanie zložky public ktorá obsahuje súbor index.html. Nasleduje

požiadavka na získanie dát z databáze. Získavajú sa len dáta ktoré spĺňajú podmienku WHERE, čiže len tie ktoré sú v rozmedzí zadaného dátumu a času od a do. Ďalej sa vytvárajú skupiny podľa id senzora. Vytvorí sa mapa údajov pre každý senzor samostatne podľa jeho id. Potom sa vytvorí názov do legendy grafu podľa id senzoru. Tým sa ukončí funkcia pre získavanie dát z sqlite databáze. Ďalej sa spustí server a vypíše sa nameraná teplota do konzole. Potom je tam funkcia na otvorenie sqlite databáze, ak žiadna neexistuje tak vytvorí novú. Ďalej vytvorí tabuľku v databázy, kde sa ukladajú záznami zo senzorov. Ešte sa musí určiť serial port z ktorého prichádzajú dáta od Arduina. Dáta prichádzaju v hexadecimálnych hodnotách a preto ich treba previesť na string. Nakoniec sa už len uložia dáta do sqlite databáze. Ukladá sa dátum,id senzora a nameraná hodnota teploty.

```
const SerialPort = require('serialport'); //požiadavka na serial port
const sqlite3 = require('sqlite3'); // požiadavka na databázu
const express = require('express'); // požiadavka na http stránku
const _ = require('underscore'); // požiadavka na skupinovanie dát na pole

const app = express(); //zavolanie aplikácie na internet
const SERVER_PORT = 3000; // server port

// používanie zložky public ktorá obsahuje index.htm
app.use(express.static('public'));

//zakladna cesta ktorá načíta index.html po zadaní príslušného portu
app.get('/', (req, res) => {
    res.sendFile(__dirname + 'index.html');
});
//požiadavka na získanie dát
app.get('/data', (req, res) => {
    const date_from = new Date(req.query.from).toISOString();
    const date_to = new Date(req.query.to).toISOString();
    console.log(date_from, date_to);

// požiadavok na databázu, ktorý vráti všetky záznamy, ktoré splňajú
//podmienku WHERE
    db.all(`SELECT time as t, sensor_id, value as y FROM sensors WHERE time > ?
         AND time < ?`, [date_from, date_to], (err, rows) => {
        if (err) {
            throw err;
        }
```

```
//vytvorenie skupiny, kde id skupiny je kľúč a hodnoty sú zánzamy s daným
//id
        const groups = _.groupBy(rows, (row) => row.sensor_id);
// groups je pole, kde každý prvok poľa je ďalšie pole obsahujúce záznamy
//pre každý senzor
        const dataset_group = _.toArray(groups).map((group) => {
            const grp = {};
// z prvého prvku zistíme id a vytvoríme názov, ktorý sa bude zobrazovať
//ako legenda grafu
            grp.label = `Sensor ${group[0].sensor_id}`;
            group.forEach((line) => {
                delete line.sensor_id;
            });

            grp.data = group;

            return grp;
        });
        res.json(dataset_group)
    });
});
//spustí server a vypisuje do konzole nameranú teplotu
app.listen(SERVER_PORT, () => console.log(`Sensor app listening on port ${
    SERVER_PORT}!`));
//otvára databázu s názvom sensors.sb, ak neexistuje vytvorí novú
let db = new sqlite3.Database('./sensors.db', (err) => {
    if (err) {
        console.error(err.message);
    } else {
        console.log('Connected to the database');
    }
});
//vytvorí sa tabuľka pre ukladanie záznamov so senzorov
//timestamp dátovy typ pre ukladanie dátumu
db.run(`CREATE TABLE IF NOT EXISTS sensors (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            time timestamp NOT NULL,
            sensor_id INTEGER NOT NULL,
            value DOUBLE NOT NULL)`);
```

```
//serial port na ktorom je pripojené arduino
const serialPort = new SerialPort('/dev/ttyUSB0', {
    baudRate: 9600,
});

serialPort.on('data', data => {
    const date = new Date().toISOString();
//data prichádzajú v hexa buffery, toString() ich prevedie na string
    const line = data.toString();
    console.log(`[${date}] ${line}`);

    const sensor_values_raw = line.split(" ");
    sensor_values_raw.forEach((sensor) => {
        const parts = sensor.split(":");
        const sensor_id = parts[0];
        const sensor_value = parts[1];
//uloží zánzam do databáze
        db.run(`INSERT INTO sensors (time, sensor_id, value)
                    VALUES(?, ?, ?)`,
            [date, sensor_id, sensor_value]
        )
    })
});
```

Výpis 6: Skript pre automatizované meranie

4.4 Aplikácia na servery

Pre vykresľovanie grafu na servery je potrebné vytvoriť webovú stránku, ktorá toho bude schopná. Tento kód vytvorí pomocou html a javascriptu vzhľad aj funkcionalitu webstránky. Stránka obsahuje tlačítko na potvrdenie, tlačítko na reset priblíženia, box na zadávanie času od kedy chceme vidieť nameraný hodnoty a box na zadávanie do kedy chceme vidieť namerané hodnoty. Taktiež samozrejme vykresľuje samotný graf s osami pre všetky senzory, od ktorých Arduino získava údaje. Ďalej sa dá tento graf priblížiť alebo oddialiť, alebo ukázať presnú hodnotu nameranej teploty po ukázani na konkrétny bod grafu. Taktiež sa dajú skryť alebo naopak odkryť čiary grafu podľa potreby.

Na začiatku sa definuje samotný dokument a používaný jazyk. Potom sa určí názov stránky. Ďalej sa nastaví tlačítko na spustenie vykreslenia grafu, tlačítko na resetovanie priblíženia grafu, box na napísanie dátumu a času od kedy sa majú ukázať uložené dáta v grafe a box na napísanie

dátumu a času do kedy sa majú ukázať uložené dáta. Nasleduje samotné získanie týchto dvoch dátumov a časov z boxov. Na začiatku sa nastaví čas v prvom boxe o 30 minút menej, ako v druhom. Nasleduje funkcia na samotné vykreslenie grafu, kde sa určia parametre ako od kedy do kedy sa má graf vykresliť, o aký typ grafu pôjde, názov grafu, pomenovanie obidvoch ôs, nastavenie pohybu do strán a nakoniec nastavenie zväčšovania či zmenšovania grafu. Ďalej je tam funkcia na získavanie náhodnej farby, ktorú priradí daným čiaram grafu. Nakoniec je tam ešte funkcia na formátovanie času do potrebnej podoby.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Arduino temperature</title>
</head>
<body>
<button onclick="getData()">Submit</button>//Tlačítko potrvdiť

<label for="dateFrom">Time from</label>//Čas od kedy
<input id="dateFrom" type="datetime-local"/>

<label for="dateTo">Time to</label>//Čas do kedy
<input id="dateTo" type="datetime-local"/>
//Tlačítko na reset
<button id="resetZoom" onclick="window.chart.resetZoom()">Reset zoom</button>

<div style="width: 75\%;margin:0 auto;">
    <canvas id="myChart"></canvas>
</div>

<script>
    let ctx;
    let date_picker_from;
    let date_picker_to;

    window.onload = function () {
//Získanie dát od a do určeného času do grafu
        ctx = document.getElementById("myChart").getContext("2d");
        date_picker_from = document.getElementById("dateFrom");
        date_picker_to = document.getElementById("dateTo");
```

```
//Nastavenie aktuálneho času a 30 minút dozadu na začiatku
        const current_time = new Date();
        const before_30_minutes = new Date();
        before_30_minutes.setMinutes(before_30_minutes.getMinutes() - 30);

        date_picker_from.value = formattedDateTime(before_30_minutes);
        date_picker_to.value = formattedDateTime(current_time);
    };

//funkcia na vykreslenie dát do grafu
    function getData() {
        const xmlHttp = new XMLHttpRequest();
        xmlHttp.onreadystatechange = function () {
            if (xmlHttp.readyState === 4&& xmlHttp.status === 200) {
                const data = JSON.parse(xmlHttp.responseText);

                data.forEach((dataset) => {
                    dataset.fill = false;
                    dataset.borderColor = dynamicColors();
                });
                drawChart(data)
            }
        };
//Nastavenie zadaného času od a do
        const date_from = date_picker_from.value;
        const date_to = date_picker_to.value;
//Vykreslenie dát podľa zadaného času
        const query_params = `?from=${date_from}&to=${date_to}`;

        xmlHttp.open("GET", `http://localhost:3000/data${query_params}`, true);
        xmlHttp.send(null);
    }
//Funkcia na vykreslenie grafu
    function drawChart(data) {
        if (window.chart) {
            window.chart.destroy();
        }
        window.chart = new Chart(ctx, {
            type: 'line',//vykresľovanie čiarového grafu
```

```
            data: {
                datasets: data,
            },
            options: {
                elements: {
                    line: {
                        tension: 0
                    }
                },
                animation: {
                    duration: 0
                },
                hover: {
                    animationDuration: 0
                },
                responsiveAnimationDuration: 0,
                responsive: true,
//Názov grafu
                title: {
                    display: true,
                    text: "Graph measuring temperature"
                },
                tooltips: {
                    callbacks: {
                        label: function (tooltipItem) {
                            return `${tooltipItem.yLabel} degree C`;
                        }
                    }
                },
//Názvy osý grafu
                scales: {
                    xAxes: [{
                        type: "time",
                        scaleLabel: {
                            display: true,
                            labelString: 'Time[s]'
                        }
                    }],
```

```
                yAxes: [{
                    scaleLabel: {
                        display: true,
                        labelString: 'Temperature[C]'
                    },
                }]
            },
//Pohyb ťahaním do bokov grafu
            pan: {
                enabled: true,
                mode: "x",
                speed: 10,
                threshold: 10
            },
//Priblíženie a oddialenie grafu
            zoom: {
                enabled: true,
                drag: false,
                mode: "xy",
                limits: {
                    max: 10,
                    min: 0.5
                }
            }
        }
    })
}

//Vygenerovanie náhodnej farby pre čiaru grafu
function dynamicColors() {
    const r = Math.floor(Math.random() * 255);
    const g = Math.floor(Math.random() * 255);
    const b = Math.floor(Math.random() * 255);
    return "rgb(" + r + "," + g + "," + b + ")";
}
//Pridanie 0 ku časom, ktoré sú menšie ako 10
function pad(n) {
    return n < 10 ? '0' + n : n
}
```

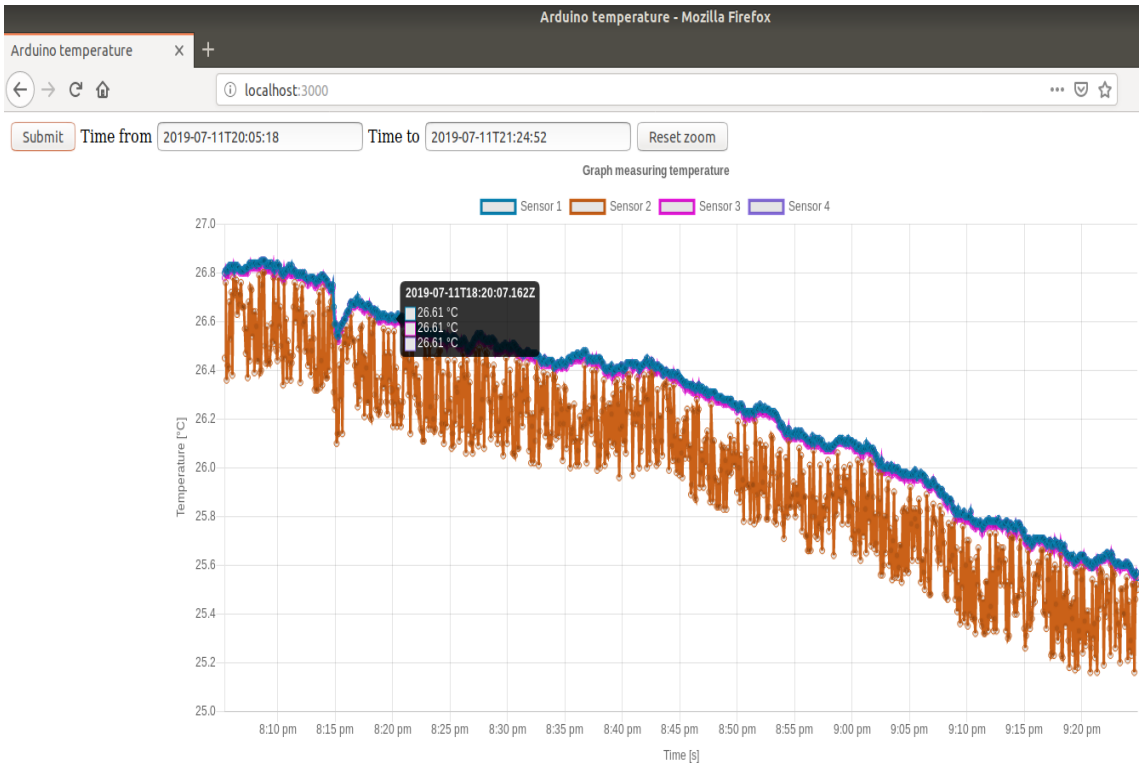
```
//Formátovanie času
    function formattedDateTime(date) {
        return `${date.getFullYear()}-${pad(date.getMonth() + 1)}-${pad(date.
            getDate())}T${pad(date.getHours())}${pad(date.getMinutes())}:${pad(
            date.getSeconds())}`
    }


</script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.8.0/Chart.bundle
    .min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/hammerjs@2.0.8"></script>
<script src="https://cdn.jsdelivr.net/npm/chartjs-plugin-zoom@0.7.2"></script>
</body>
```

Výpis 7: Aplikácia na localhoste

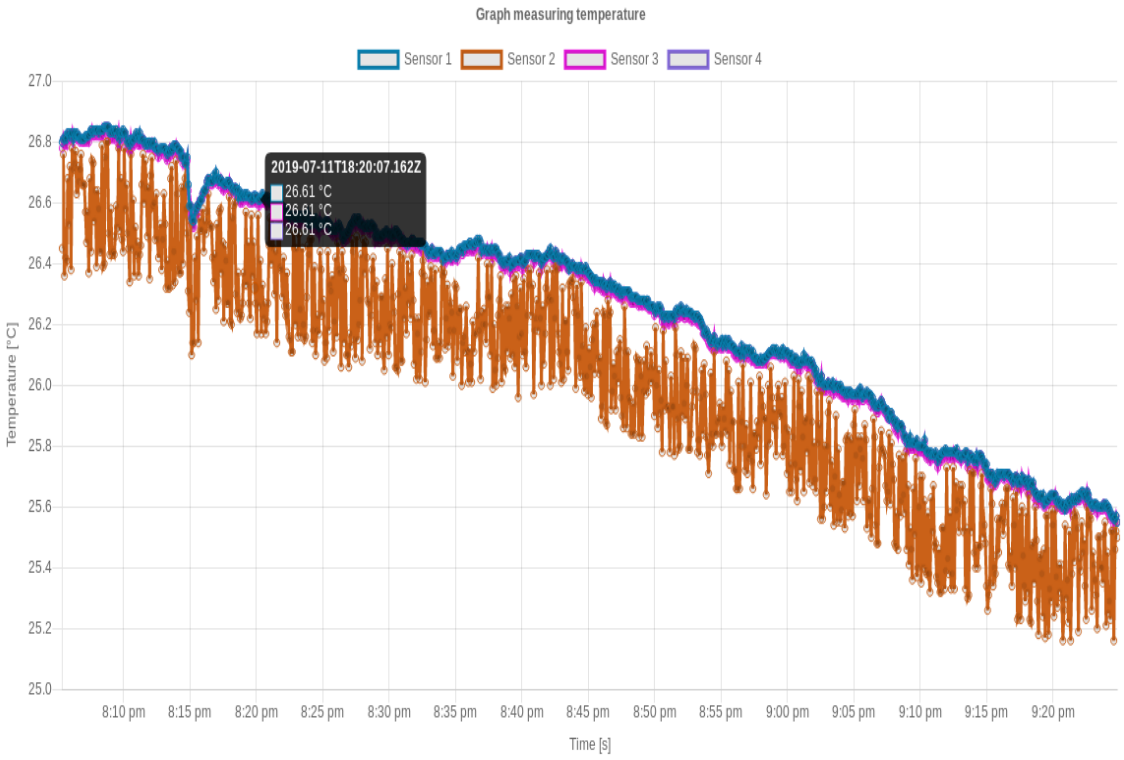
5 Grafy z meraní

Na začiatok je tu screenshot, na ktorom je vidno celú aplikáciu spustenú na localhoste s vykresleným grafom. Meranie spracované do grafu trvalo od 11.7.2019 o 20:05 až do 11.7.2019 o 21:25 minúte. Merala sa teplota so všetkými štyrmi senzormi naraz. Tlačítko Submit slúži ku vykresleniu nového grafu v zadanom dátume a čase. Ypsylónová os označuje nameranú teplotu grafu a Xová os označuje čas, v ktorom bola táto hodnota odmeraná. Senzory sú usporiadané podľa id, ktoré sa zadáva pri nahrávaní kódu na Arduino. V tomto prípade Senzor 1 je senzor BMP280, Senzor 2 je termistor, Senzor 3 je teplotné čidlo DS18B20 a Senzor 4 je senzor HTU21D.



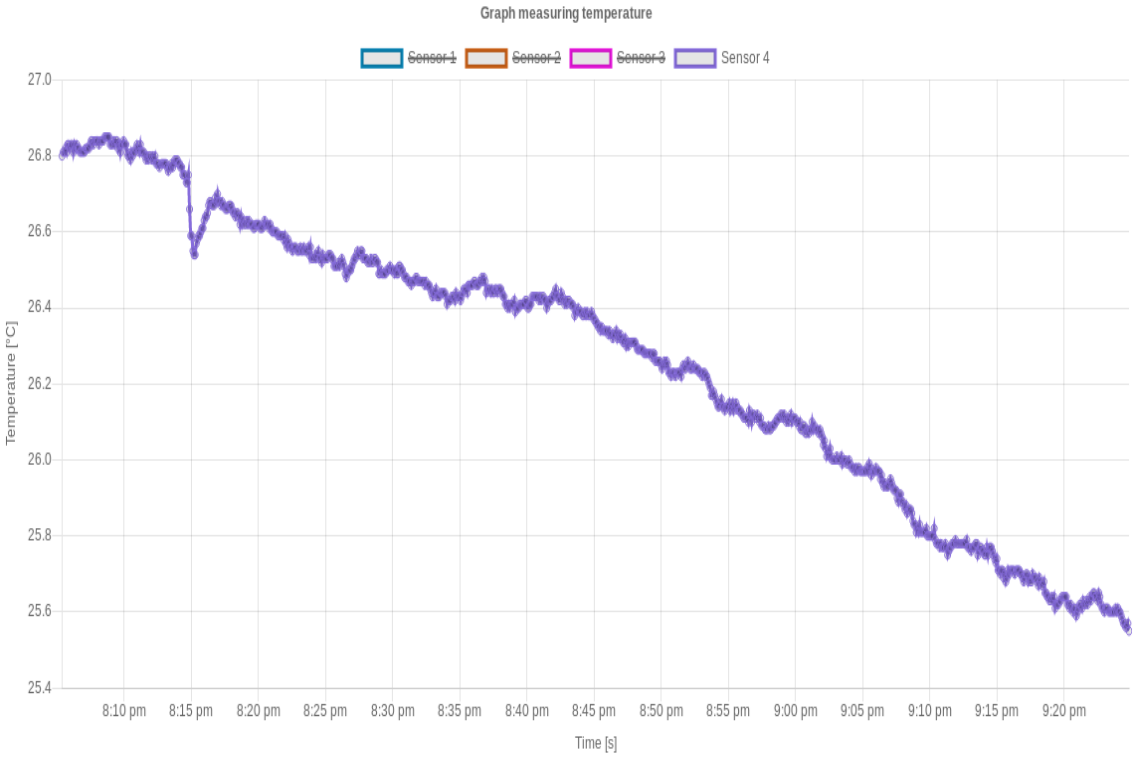
Obrázek 8: Výsledná aplikácia

Na ďalšom obrázku grafu je vidno výslednú aplikáciu po nameraní dát. V tomto konkrétnom prípade ide o meranie teploty pomocou štyroch senzorov zapojených naraz. Meranie spracované do grafu je rovnaké ako na prvom obrázku celej výslednej aplikácie. Senzory HTU21D a BMP280 merali teploty úplne rovnako preto sa tieto dve čiary prekrývajú. Senzor DS18B20 meral teplotu veľmi podobne s odchylkou maximálne 0,2 stupňa Celzia.Čo sa týka termistoru jeho meranie nebolo úplne presné. Na grafe vidno veľké výkyvy teplôt, aj keď len vrámci jedného stupňa celzia. Toto konkrétne meranie bolo odmerané vonku v podvečer leta takže na neho menej vplývali vonkajšie parametre, ako sú vietor alebo slnko.



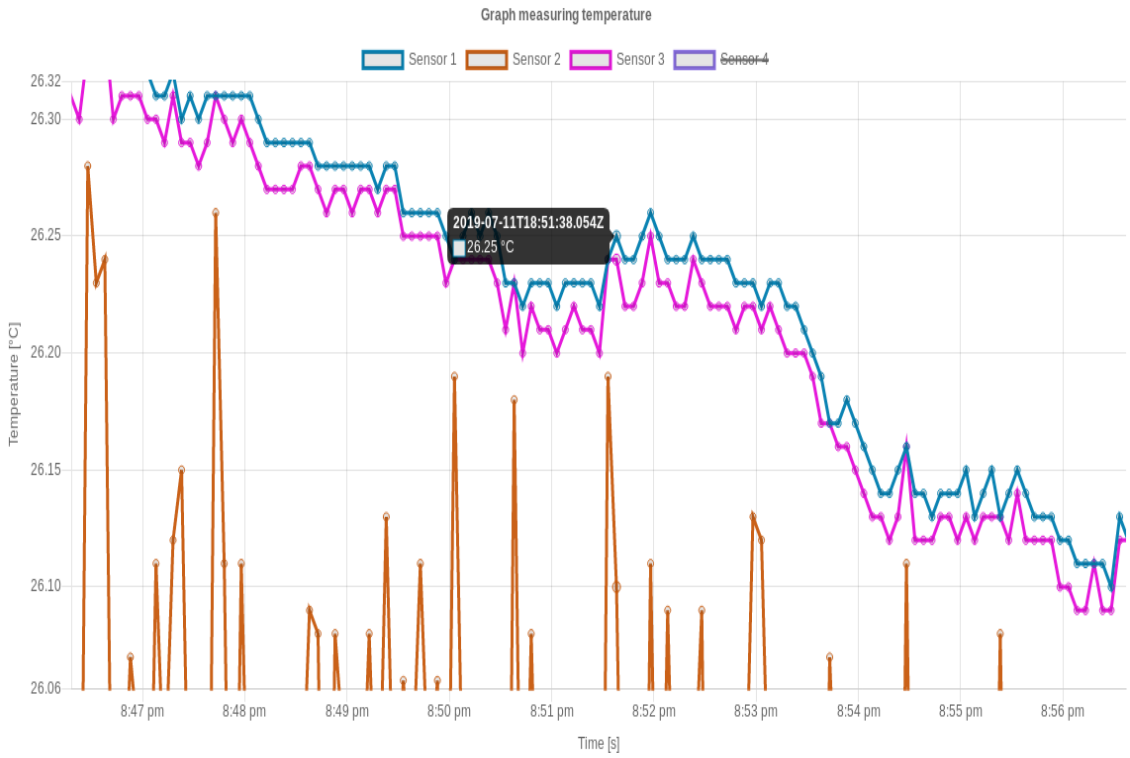
Obrázek 9: Graf štyroch senzorov

V ďalšom grafe je to isté meranie, ako v predošlom s tým rozdielom že sú vypnuté body ktoré ukazujú všetky čiary. Spustená je len čiara pre senzor BMP208. Tieto čiary v grafe sa dajú kedykoľvek zapínať a vypínať pomocou tlačidiel s názvom senzoru podľa jeho id v legende nad grafom. Nápis pre senzory, ktorých zobrazovanie čiar je vypnuté sú potom prešKrtnuté.



Obrázek 10: Graf jedného senzoru

Na poslednom grafe je stále to isté meranie akurát je graf priblížený na konkrétny bod nameranej teploty. Po ukázaní na tento bod sa zobrazí legenda všetkých grafov , ktoré v tomto bode majú nameranú hodnotu a taktiež dátum a čas kedy presne táto hodnota bola odmeraná. Približovanie aj oddiaľovanie je možné pomocou kolieska na myši. Ak je potrebné vrátiť sa do pôvodného priblíženia, stačí stlačiť tlačidlo Reset zoom.



Obrázek 11: Priblížený graf

6 Záver

Táto bakalárska práca obsahuje teoretický popis štyroch použitých senzorov na meranie teploty a taktiež schémy ich zapojenia ku Arduinu. Ďalej je tu popis všetkých použitých zberníc na prenos dát či už medzi senzorom a Arduinom, alebo aj medzi Arduinom a počítačom. Následne sú tu opísané kódy potrebné ku spusteniu merania teploty s týmito senzormi. Tieto kódy sa musia nahrať na Arduino ku spusteniu samotného merania teploty. Po nahraní kódov je tu spísaný skript v nodejs spustiteľný z terminálu na operačnom systéme Linux, ktorý prijíma dáta zo sériovej linky a ukladá ich do databáze v sqlite. Taktiež vytvára webovú stránku na servery, kde sa pomocou html a javascriptu vytvorí aplikácia, ktorá vytvára graf z nameraných hodnôt, ktoré sú uložené v sqlite databáze. Graf obsahuje všetky namerané údaje rozdelené podľa jednotlivých senzorov v danom čase. Dá sa taktiež vybrať čas od kedy do kedy sa majú zobrazovať namerané hodnoty v grafe. Graf sa dá priblížiť alebo oddialiť podľa potreby, a taktiež posúvať. Výsledná aplikácia môže byť použitá ku pozorovaniu teploty v akúkoľvek dobu.

Čo sa týka senzorov tak vrámci meraní všetky senzory merali teplotu veľmi presne a podobne. Rozdiely v meraných teplotách boli maximálne 0,3 stupňa Celzia. Na meranie vplývali vonkajšie parametre ako napríklad teplota počítača, vietor alebo slnko. Pri nízkych teplotách senzor DS18B20 aj napriek tomu, že bol vo vodotesnom obale sa začal odchyľovať od ostatných senzov. Týkalo sa to hlavne meraní s teplotou pod 5 stupňov Celzia. Termistor mimo obalu meral teplotu veľmi nepresne, odchylky boli niekedy až jeden stupeň Celzia. Po vložení do ochranného obalu začal merať presnejšie s vychýlkami maximálne 0,5 stupňa Celzia.

Skript v nodejs bol otestovaný v operačnom systéme Linux aj v operačnom systéme Windows. V oboch prípadoch fungoval bez problémov. Je schopný prispôsobiť sa počtu zapojených senzorov, podľa toho koľko údajov posiela Arduino po sériovej linke. Každému senzoru automaticky pridá identifikačné číslo, pod ktorým ukladá dáta do databázy a taktiež zobrazuje údaje na grafe. Danú aplikáciu na vykresľovanie grafu spúšťa na porte 3000, takže je možné sa ku nemu pripojiť aj pomocou internetovej siete.

Na koniec tu boli ukážky konkrétnych grafov s nameranými teplotami v určenej dobe a taktiež obrázok celej výslednej aplikácie. Všetky tieto obrázky ukážok boli popísané a vysvetlené.

Literatura

[1] Co je to Arduino? [online]. [cit. 2019-04-29]. Dostupné z: https://arduino.cz/co-je-to-arduino/

[2] GERTZ, Emily a Patrick DI JUSTO. Environmental Monitoring with Arduino [online]. United States of America: O'Reilly Media, 2012 [cit. 2019-04-29]. ISBN 978-1-449-31056-1. Dostupné z: https://alejandroquinteros.files.wordpress.com/2012/11/environmental-monitoring-with-arduino.pdf

[3] Obrázok Arduino UNO [online]. [cit. 2019-07-02]. Dostupné z: https://static4.arrow.com/-/media/arrow/images/products/1217/1217-arduino-uno-smd.jpg

[4] Měření teploty s termistorem [online]. [cit. 2019-04-29]. Dostupné z: https://navody.arduino-shop.cz/arduino-projekty/mereni-teploty-s-termistorem.html

[5] Obrázok Termistoru [online] . [cit. 2019-07-02]. Dostupné z: http://bildr.org/2012/11/thermistor-arduino/

[6] Senzor teploty a vlhkosti HTU21D I2C [online]. [cit. 2019-04-29]. Dostupné z: https://navody.arduino-shop.cz/navody-k-produktum/senzor-teploty-a-vlhkosti-htu21d-i2c.html

[7] I2C Senzor Teploty a Vlhkosti HTU21D [online]. [cit. 2019-04-29]. Dostupné z: https://www.gme.cz/i2c-senzor-teploty-a-vlhkosti-htu21d

[8] Senzor Tlaku a Teploty BMP280 [online]. [cit. 2019-04-29]. Dostupné z: https://navody.arduino-shop.cz/navody-k-produktum/senzor-tlaku-a-teploty-bmp280.html

[9] Obrázok senzora BMP280 [online]. [cit. 2019-07-02]. Dostupné z: https://cdn.myshoptet.com/usr/www.vokolo.cz/user/shop/detail_alt_1/19869_bme280.jpg?5bcdbc85

[10] Teplotný senzor DS18B20 [online]. [cit. 2019-04-29]. Dostupné z: https://navody.arduino-shop.cz/navody-k-produktum/teplotni-senzor-ds18b20.html

[11] Obrázok senzora DS18B20 [online]. [cit. 2019-07-02]. Dostupné z: https://arduino-shop.cz/photos/produkty/d/2/2029.jpg?m=1508313854

[12] Schéma zapojenia senzora DS18B20 [online]. [cit. 2019-07-02]. Dostupné z: /https://www.tweaking4all.com/wp-content/uploads/2014/03/ds18b20-arduino-connected.jpg

[13] Sběrnice 1-Wire^{TM} [online]. [cit. 2019-04-29]. Dostupné z:https://vyvoj.hw.cz/navrh-obvodu/rozhrani/sbernice-1-wiretm.html

[14] Stručný popis sběrnice I2C [online]. [cit. 2019-04-29]. Dostupné z: https://vyvoj.hw.cz/navrh-obvodu/strucny-popis-sbernice-i2c-a-jeji-prakticke-vyuziti-k-pripojeni-externi

[15] Komunikační protokol universální sériové sběrnice [online]. [cit. 2019-04-29]. Dostupné z: https://www.root.cz/clanky/komunikacni-protokol-universalni-seriove-sbernice/

[16] USB - univerzálna sériová zbernica [online]. [cit. 2019-04-29]. Dostupné z: https://shopdelta.eu/usb-univerzalna-seriova-zbernica_l9_aid745.html

[17] I2C skener adries [online]. [cit. 2019-07-10]. Dostupné z: https://navody.arduino-shop.cz/navody-k-produktum/i2c-skener.html